命令模式

ROUTINE name [([[VAR] parameter];...)]

Defines a programming routine with a parenthesized list of parameters.

定义带有括号的参数列表的编程例程。

ROUTINE [/LIST[:pattern]]

Lists the names of all matching routines (screen only).

列出所有匹配例程的名称（仅限屏幕）。

参数说明

Name

The routine name (no embedded spaces or other separators).

例程名称（无嵌入空格或其他分隔符）。

Parameter

One of a series of semicolon-separated parameter variable names (which should not contain separators or periods, nor begin with a digit). Each parameter declares a routine-local variable that is assigned the corresponding value from the parameter list when the routine is called. If preceded by the keyword VAR, the parameter refers to a variable (not a value) passed by the caller, so that any value assigned by the routine to that parameter is passed back to the caller's variable when the routine returns.

一系列以分号分隔的参数变量名称之一（不应包含分隔符或句点，也不应以数字开头）。每个参数声明一个例程局部变量，在调用例程时，从参数列表中分配相应的值。如果前面有关键字 VAR，则参数引用调用方传递的变量（而不是值），以便当例程返回时，例程分配给该参数的任何值都将传递回调用方的变量。

/LIST [:pattern]

Restricts list to only names matching pattern, which may include wildcard * and ? characters.

将列表限制为仅名称匹配模式，其中可能包括通配符 * 和 ？字符。

Operation

操作

ROUTINE definition establishes a new routine name and begins storing subsequent lines of command input as the body of the routine, replacing any old definition. All subsequent input is taken as part of the routine definition until a line is encountered comprised of the keyword END followed by the routine name. Hence a ROUTINE definition has the following form:

ROUTINE 定义建立一个新的例程名称，并开始将后续命令输入行存储为例程的主体，替换任何旧定义。所有后续输入都作为例程定义的一部分，直到遇到由关键字 END 后跟例程名称组成的行。因此，ROUTINE 定义具有以下形式：

```
ROUTINE XXX
   ROUTINE YYY (source)
      MESSAGE "This is routine YYY speaking from {source}.
   END YYY
   YYY ("XXX")
END XXX
```

A routine may be called directly using its name followed by a parenthesized comma-separated list of parameters (as seen in XXX's call of YYY above). Even if a routine has no parameters, calling it still requires empty parentheses. Thus routine XXX is called as follows.

可以使用例程的名称直接调用例程，后跟括号中的逗号分隔的参数列表（如上面 XXX 对 YYY 的调用所示）。即使例程没有参数，调用它仍然需要空括号。因此，例程 XXX 的调用如下。

```
XXX ()
```

A routine may also assign a value to its name that will be returned to its caller. This value can then be assigned directly to a variable when the routine is called. For example,

例程还可以为其名称分配一个值，该值将返回给其调用方。然后，可以在调用例程时将此值直接分配给变量。例如:

```
ROUTINE PRODUCT (A; B)
    PRODUCT := A * B
END PRODUCT
VARIABLE X
X := PRODUCT(3,4) `X now has the value 12
```

Routines can also return their values anywhere using braced expressions. For example.

例程还可以使用大括号表达式在任意位置返回其值。例如。

```
MESSAGE {PRODUCT(3,4)} `product value 12 is displayed on the screen
```

All names appearing in a routine refer to local variables, macros, templates, and subroutines declared within the routine, unless prefaced by a period "." to refer to global names (such as system variables). Module names can be accessed by prefacing with "^." and names declared in any calling routine can be accessed by prefacing with "^". For example:

例程中出现的所有名称都引用例程中声明的局部变量、宏、模板和子例程，除非前面以句点“.”开头以引用全局名称（如系统变量）。模块名称可以通过以“^.”开头来访问，在任何调用例程中声明的名称可以通过以“^”开头来访问。例如：

```
ROUTINE OUTER
    VARIABLE HEEL = 1
    ROUTINE INNER
        VARIABLE HEEL = 2
        MESSAGE INNER HEEL: {HEEL}
```

```
        MESSAGE OUTER HEEL: {^HEEL}
        MESSAGE SYSTEM VARIABLE HEEL: {.HEEL}
    END INNER
    INNER()
END OUTER
```

IF statements operate differently within routines, using a multi-line block separated by ELSIF and ELSE lines and terminated by an END line, instead of being all contained in a single line. Hence an IF block appearing within a routine has the following form:

IF 语句在例程中的操作方式不同，使用由 ELSIF 和 ELSE 行分隔并由 END 行终止的多行块，而不是全部包含在一行中。因此，例程中出现的 IF 块具有以下形式：

```
IF condition THEN
    command lines to do when IF condition is true
[ ELSIF condition THEN
    command lines to do when ELSIF condition is first to be true ]...
[ ELSE
    command lines to do when no IF or ELSIF condition is true ]
END
```

Note that ELSIF and ELSE sections are optional, and multiple ELSIF sections can appear in the same IF block. IF blocks can be nested to as many levels as desired.

请注意，ELSIF 和 ELSE 部分是可选的，多个 ELSIF 部分可以出现在同一个 IF 块中。IF 块可以根据需要嵌套到任意多个级别。

Conditions like operand1 = operand2 use numeric comparison instead of character-bycharacter comparison unless at least one operand is surrounded by quotes. Unquoted operands are automatically evaluated as expressions as if enclosed by braces. For example.

像 operand1 = operand2 这样的条件使用数字比较而不是逐个字符比较，除非至少有一个操作数被引号括起来。不带引号的操作数会自动计算为表达式，就像用大括号括起来一样。例如：

```
IF POS < LEN - 1 THEN
    IF LOGGING = "TRUE" THEN
        MESSAGE NOT YET AT PENULTIMATE POSITION
    END
END
```

LOOP blocks are provided within routines to skip back to the top of the LOOP when its matching END line is reached, only exiting the LOOP when an EXIT line is encountered. Note that LOOP and IF blocks can be nested together to any level, so an EXIT line gets out of the innermost active LOOP block, skipping through any intervening IF blocks. For example.

例程中提供了 LOOP 块，以便在到达其匹配的 END 行时跳回 LOOP 的顶部，仅在遇到 EXIT 行时才退出 LOOP。请注意，LOOP 和 IF 块可以嵌套在一起到任何级别，因此 EXIT 线从最里面的活动 LOOP 块中出来，跳过任何中间的 IF 块。例如。

```
VARIABLE I=1, N=10
LOOP
    IF I>N THEN
        EXIT
    ELSE
        MESSAGE {I}
    END
    I:=I+1
END
```

Routine processing is terminated and all its local storage released when the end of a routine is reached or an EXIT command (including parameters or outside any LOOP) is encountered.

当到达例程结束或遇到 EXIT 命令（包括参数或任何 LOOP 之外）时，例程处理将终止并释放其所有本地存储。

Display Output

输出显示

All existing routine names (on the screen only) if no parameters are present.

所有现有例程名称（仅在屏幕上），如果不存在参数。

Nondisplay Output:.

无输出显示：

none

无

Examples

样例

Defining a routine to return the factorial (n!) of a non-negative integer:

定义一个例程以返回非负整数的阶乘 （n！）

```
ROUTINE FACTORIAL (N)
    IF N<>TRUNC(N) THEN
        ME "{N} must be an integer.
        EXIT
    ELSIF N<0 THEN
```

```
        ME "{N} must not be negative.
        EXIT
    END
    VARIABLE F={N}
    LOOP
        N:=N-1
        IF N<=1 THEN
            FACTORIAL:=F
            EXIT
        END
        F:=F*N
    END
END FACTORIAL
ME {FACTORIAL(5)} `displays 5! = 120
```